SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

HTTP-PALVELU PDF-TIEDOSTOJEN KÄSITTELYYN JA MUOKKAAMISEEN

TEKIJÄ/T: Moona Partanen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Moona Partanen	
Työn nimi HTTP-palvelu pdf-tiedostojen käsittelyyn ja muokkaamiseen	
Päiväys 8.4.2018	Sivumäärä/Liitteet 25/0
Ohjaaja(t) Lehtori Jussi Koistinen, Lehtori Jukka Kinnunen	
Toimeksiantaja/Yhteistyökumppani(t) Ropo Capital	

Tiivistelmä

Opinnäytetyön aiheena oli suunnitella ja toteuttaa HTTP-palvelu, joka käsittelee pdf-tiedostoja. Opinnäytetyön toimeksiantajana toimi Ropo Capital. Palvelulla oli todellinen tarve, sillä Ropo Capitalilla käytössä oleva ohjelmisto pdf-tiedostojen käsittelyyn on poistumassa käytöstä ja sen tilalle tarvittiin uusi palvelu. Uuden palvelun toimintoja ovat pdf-tiedostojen pilkkominen pienempiin osiin, Postin palautuskoodin lisääminen pdf-tiedoston halutuille sivuille, useamman pdf-tiedoston yhdistely sekä tiedostojen konvertointi.

Palvelu toteutettiin Java-ohjelmointikielellä. Palvelu koostuu useasta mikropalvelusta, joita voidaan kutsua eri asiakasohjelmista HTTP-rajapintaa hyödyntäen.

Lopputuotteena syntyi HTTP-palvelu pdf-tiedostojen käsittelyyn ja muokkaamiseen. Palvelu otettiin käyttöön Ropo Capitalilla toteutusvaiheen jälkeen.

Avainsanat
Java, http-palvelu, REST

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Moona Partanen			
Title of Thesis Http Service for Processing and Editing Pdf Files			
Date	8th April 2018	Pages/Appendices	25/0
Supervisor(s) Mr. Jussi Koistinen, Lecturer and Mr. Jukka Kinnunen, Lectorer			
Client Organisation /Partners Ropo Capital			

Abstract

The purpose of this thesis was to design and develop an HTTP service that processes pdf files. The thesis was commissioned by Ropo Capital. There was a real demand for the service because the software that is in use in Ropo Capital for the processing of pdf files will be abandoned and that is the reason why the new service is needed. The features which are included in the new service are splitting pdf files into smaller parts, adding the return code of Posti to selected pages in a pdf file, combining multiple pdf files and converting files.

The service was developed with the Java programming language. The service consists of several micro services which can be called from different client programs using the HTTP protocol.

The result of this thesis was the HTTP service for processing and editing pdf files. The service was deployed at Ropo Capital after the development phase.

Keywords
Java, http service, REST

ESIPUHE

Haluaisin kiittää koko Ropo Capitalin kehitystiimiä mahdollisuudesta tehdä mielenkiintoinen opinnäytetyö yritykselle. Erityiskiitokset haluan antaa asiantuntijoille Janne Peltoselle, Janne Huttuselle, Jouko Pelkoselle, Mika Saarelle sekä Simo Hiltuselle, joilta sain hyviä vinkkejä opinnäytetyön toteutusvaiheessa.

Haluan kiittää myös yrityksessä toimivia asiantuntijoita Janne Martikaista, joka varsinkin opinnäytetyöni alkuvaiheilla antoi neuvoja ja järjesti opinnäytetyön aiheen suunnittelupalaverin kehitystiimin kanssa sekä Vesa Jääskeläistä, joka on kannustanut kovasti opinnäytetyön etenemisessä.

Kiitokset myös opinnäytetyöni ohjaajalle, lehtori Jussi Koistiselle, jolta sain palautetta ja ohjeistusta aina tarvittaessa.

Viimeisenä haluan vielä kiittää kavereita ja avopuolisoani Eetua, joilta olen saanut paljon tukea ja kannustusta opinnäytetyötä tehdessäni.

Kuopiossa 8.4.2018
Moona Partanen

TERMIT JA LYHENTEET

Java-ohjelmointikieli on Sun Microsystemsin 1990-luvun alussa kehittämä olio-ohjelmointikieli

Maven on Java-koodin kääntämistä helpottava komentopohjainen ohjelma

Skeema (engl. schema) on W3C: n standardoima teknologia, jolla voidaan kuvata XML-dokumenttien rakenne

Kirjastot ovat kokoelmia, aliohjelmia, ja/tai luokkia, joita käytetään ohjelmistojen kehittämisessä

HTTP eli Hypertext Transfer Protocol

Moduuli tarkoittaa ohjelmiston itsenäistä osaa, jolla on oma toiminallinen tehtävänsä

JAR-paketti (Java Archive) on tiedostomuoto, jota käytetään Java-tiedostojen ja ohjelman resurssien yhdistämisessä yhteen jakelupakettiin.

XML (Extensible Markup Language) on yksinkertainen ja joustava tekstimuoto, jolla on iso rooli erilaisten tietojen vaihdossa verkossa.

Client eli asiakasohjelma

SISÄLTÖ

1 JOHDANTO

Ropo Capital on kotimainen laskun elinkaari- ja rahoituspalveluihin erikoistunut yritys, joka kilpailee markkinoilla teknologisena edelläkävijänä. Ropo Capitalin toimintamalli pohjautuu digitalisaation etuihin ja vahvaan automaatioon. Ropo Capitalin palvelutarjonta kattaa koko laskun elinkaaren hallinnan laskujen operoinnista reskontranhoitoon, maksuvalvontaan ja rahoitukseen. Ropo Capitalilla on noin 8000 asiakasta Suomessa ja joka kuudes Suomessa välitettävä lasku kulkee Ropo Capitalin kautta.

Ropo Capitalilla on ollut käytössä GMC Inspire-ohjelmisto, mutta opinnäytetyöni aihetta miettiessä tästä järjestelmästä haluttiin luopua. Inspire kattaa useita eri toimintoja kuten esimerkiksi pdf-aineistojen käsittelyä, arkistointia, muokkausta, pilkontaa, konvertointia, paikallistulostusta sekä laskutustapahtumien kirjoittamista. Inspire-ohjelmistosta luovuttaessa piti suunnitella ja toteuttaa korvaavat palvelut, jotka tekisivat nämä toiminnot jatkossa. Opinnäytetyöni aiheeksi valikoitui pdf-tiedostojen käsittelyyn liittyvät toiminnot.

Koska Inspirestä luopumisen yhteydessä tuli esiin useampia pdf-tiedostoihin liittyviä toimintoja, sainkin opinnäytetyöni aiheeksi suunnitella ja toteuttaa palvelun, joka hoitaa jatkossa pdf-tiedostojen käsittelyä ja muokkausta Ropo Capitalin aineistokäsittelyssä. Palvelun kriteerinä oli myös se, että sitä voidaan kutsua useasta eri asiakasohjelmasta.

Opinnäytetyönäni suunnittelin ja toteutin HTTP-palvelun, joka käsittelee pdf-tiedostoja ja muokkaa niitä. Palvelussa on useampia toimintoja, kuten pdf-tiedostojen pilkonta ja Postin palautuskoodin merkkaaminen pdf-tiedostoon. Tässä raportissa kutsun palvelua sovellukseksi.

Opinnäytetyöni suunnitteluvaiheessa suunnittelin ja määrittelin sovelluksen toimintoja yhdessä muiden kehitystiimin jäsenten kanssa, mutta toteusvaihe jäi minun vastuulleni. Sain työskennellä itsenäisesti sovelluksen kehityksen ja ohjelmoinnin parissa sekä valita käytettävät tekniikat. Opinnäytetyöni lopputuotteena syntyi sovellus, joka otettiin heti käyttöön Ropo Capitalilla. Tässä raportissa kerron opinnäytetyöni työvaiheista, toteutuksesta, kehitysmenetelmistä sekä sovelluksen ominaisuuksista.

2 SOVELLUS

2.1 Tarkoitus

Opinnäytetyöni tarkoituksena oli kehittää sovellus pdf-tiedostojen käsittelyyn ja muokkaamiseen. Sovelluksella oli todellinen tarve, sillä johdannossa mainittu Inspire poistuu käytöstä Ropo Capitalilla. Sovellus siis korvaa joitain Inspiren aiemmin suorittamia toimintoja. Sovelluksen toimintoja tarvitaan Ropo Capitalin asiakkaiden aineistojen käsittelyssä. Toimintoja eli palveluja voidaan kutsua eri ohjelmista HTTP-rajapintaa hyödyntäen.

2.2 Toiminnot

Sovellus sisältää eri toimintoja ja se koostuu useasta eri mikropalvelusta. Mikropalvelulla tarkoitetaan arkkitehtuuria, jossa sovelluksen lähdekoodi on jaettu toiminnallisiin osiin eli moduuleihin (katso kohta 2.4 Rakenne). Kukin moduuli suorittaa yhtä toimintoa. (Tarvitaanko mikro-palveluja oikeasti? Verkkojulkaisu. 3.3.2018).

Sovellusta suunniteltaessa pyrittiin kokoamaan kaikki ne Inspiren toiminnot, jotka liittyvät pdf-tiedostojen käsittelyyn ja muokkaamiseen, jotta ne saataisiin kaikki samaan sovellukseen. Sovellus suorittaa seuraavia toimintoja:

1. Pdf-tiedostojen pilkonta
2. Tiedostojen konvertointi
3. Postin palautuskoodin lisääminen pdf-tiedostoon
4. Pdf-tiedostojen yhdistely

2.3 Suunnittelu

Sovelluksen suunnittelu aloitettiin aloituspalaverilla, jossa käytiin alustavasti läpi mitä ominaisuuksia ja toimintoja sovellukseen halutaan. Palaverissa mietittiin myös mahdollisia tekniikoita, joilla sovellus voitaisiin toteuttaa. Aloituspalaverin jälkeen suunnittelu jatkui määrittelypalaverilla, jossa päätettiin lopulliset tekniikat ja sovelluksen toiminnot.

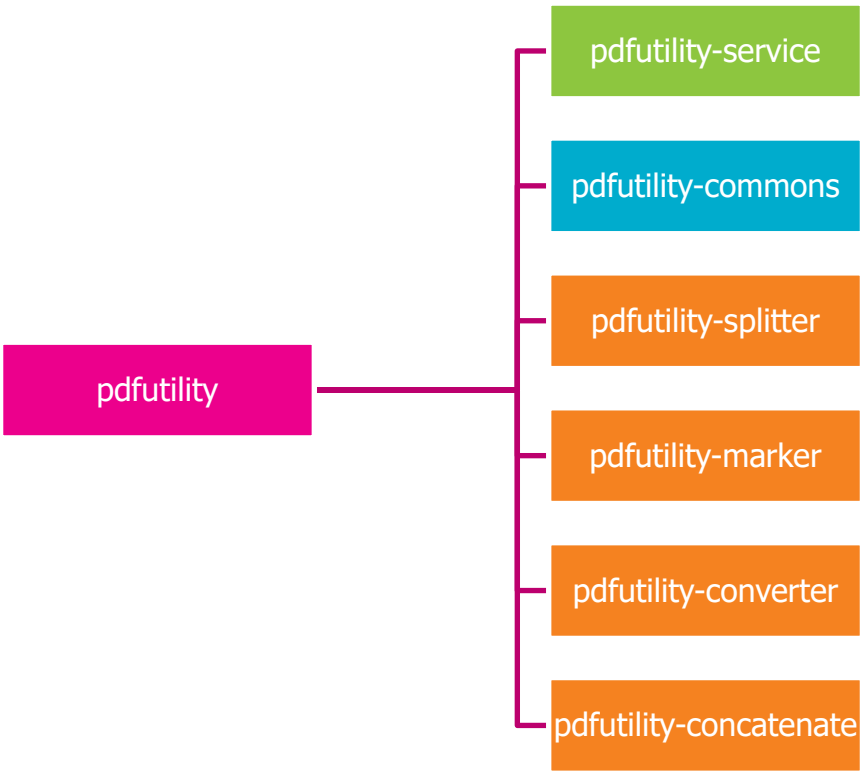
Koska sovellus tulisi Ropo Capitalin sisäiseen käyttöön ja sitä kutsuttaisiin monesta eri ohjelmasta, suunnitteluvaiheessa piti ottaa huomioon monia eri asioita. Taulukossa 1 on kuvattu suunnitteluvaiheessa huomioon otettavat asiat ja syyt, miksi asiat tuli erityisesti ottaa huomioon.

Huomioitava	Syy
Sovelluksen kuormitus	Sovellus käsittelee suuria aineistoja, joten kuormitus saattaa kasvaa isoksi, mikä voi hidastaa sovellusta jonkin verran.
Sovelluksen rakenne	Sovelluksen rakenteen ja arkkitehtuurin suunnittelu selkeäksi, jotta ylläpito ja jatkokehitys olisivat helpompaa.
Tekniikat	Käytettävä tekniikoita, jotka mahdollistavat suuren kuormituksen käsittelyn.

TAULUKKO 1. Suunnitteluvaiheessa huomioon otettavat asiat

2.4 Rakenne

Sovelluksen rakenne toteutettiin niin, että jokaisella sovelluksen toiminnoilla on oma moduulinsa. Rakenteeseen päädyttiin siksi, että selkeä rakenne helpottaa mahdollisien uusien toimintojen lisäämisen sovellukseen sekä sovelluksen ylläpidettävyys helpottuu.



KAAVIO 1. Sovelluksen rakenne

Kaaviossa 1 on kuvattu sovelluksen rakenne. Pdfutility on niin sanottu päämoduuli, joka sisältää sovelluksen toiminnallisuudet omina moduuleinaan. Moduulit voivat olla riippuvaisia toisistaan eli ne voivat käyttää toistensa luokkia. Kaaviossa 1 oranssilla värillä merkityt moduulit suorittavat erilaisia toimintoja. Toimintoja suorittavia moduuleita voidaan päivittää, ylläpitää ja lisätä helposti, koska ne ovat omia, irrallisia moduuleitaan.

2.4.1 Pdfutility-commons

Pdfutility-commons-moduuli sisältää kaikki yleiset ominaisuudet, joita voidaan tarvita myös muissa moduuleissa. Esimerkiksi niin sanotut apuluokat, jotka eivät sovi mihinkään kokonaisuuteen sisällytetään pdfutility-commons-moduuliin.

2.4.2 Pdfutility-service

Pdfutility-service-moduulissa ovat kaikki Rest-rajapintaan tarvittavat luokat ja toiminnallisuudet. Esimerkiksi Jetty-serverin (katso kohta 3.1.4. Jetty-palvelin) luokat ja handlerit sisältyvät pdfutility-service-moduuliin. Koska sovellusta käynnitettäessä kutsutaan ensimmäiseksi pdfutility-service-moduulia, tehtiin siitä JAR-paketti, joka sitten laitettiin pyörimään sekä Windows-palveluksi että AWS: lle (Katso kohdat 4.4. AWS ja 4.5. JAR-paketti Windowsin palveluksi).

2.4.3 Mikropalvelut

Kaaviossa 1 oranssilla merkityt moduulit ovat mikropalveluja, jotka suorittavat kukin yhtä toimintoa. Esimerkiksi pdfutility-splitter-moduulin tehtävä on hoitaa pdf-tiedostojen pilkkominen sovelluksessa. Moduuli sisältää siis kaikki pilkontaan tarvittavat luokat. Pdfutility-splitter on riippuvainen pdfutility-commons-moduulista, sillä yleiset toiminnallisuudet kuten esimerkiksi tiedostojen pakkaus zip-tiedostoksi tapahtuu pdfutility-commons-moduulissa.

Yllämainitun tapaan myös muut kaaviossa 1 oranssilla merkityt moduulit suorittavat omia toimintojaan. Pdfutility-marker-moduuli hoitaa postin palautuskoodin lisäämisen pdf-tiedostoon ja se sisältää kaikki merkkaamiseen tarvittavat luokat ja ohjelmakoodit. Pdfutility-converter-moduuli sisältää kaikki konvertoinnissa tarvittavat luokat, kun taas pdfutility-concatenate-moduulissa on yhdistelyn suorittavat ohjelmakoodit.

3 KÄYTETYT TEKNIIKAT JA KEHITYSMENETELMÄT

3.1 Tekniikoiden ja kehitymenetelmien valitseminen

Koska toteuttamani sovellus on erillinen palvelunsa ja sain kehittää sitä itsenäisesti, sen toteutukseen käytetyt tekniikat olivat vapaasti valittavissa. Tekniikoita valittaessa kuitenkin piti ottaa huomioon se, että valitut tekniikat mahdollistavat sovelluksen suuren kuormituksen. Sain tekniikoiden valitsemiseen neuvoa ja vinkkejä muilta kehitystiimin jäseniltä.

Kehitysmenetelmät määräytyivät Ropo Capitalin käytäntöjen mukaan. Esimerkiksi Scrumia ja GITiä käytetään kaikissa projekteissa. Myös yksikkötestaus ja testien tekeminen on yleinen tapa Ropo Capitalilla.

3.1.1 Java

Java on Sun Microsystemsin kehittämä olio-ohjelmointikieli, joka on C++: n pohjalta rakennettu. Javan tyypillisiä piirteitä ovat muun muassa puhdas olioperustaisuus ja vahva tyypitys. Javan etuihin kuuluu erittäin kattava standardikirjasto sekä muut saatavilla olevat kirjastot. Sovelluksen ohjelmointikieleksi valittiin Java sen laajojen kirjastojen takia. (VESTERHOLM, Mika ja KYPPÖ, Jorma. 2015. 3.3.2018).

3.1.2 Eclipse Oxygen

Eclipse on ohjelmointiympäristö, joka tukee Java-ohjelmointikieltä. Sovelluksen kehitykseen ohjelmistoympäristöksi valittiin Eclipse Oxygen, koska se oli opinnäytetyöni tekemishetkellä uusin Eclipsen versio. (Eclipse (IDE), Wikipedia. 3.3.)

3.1.3 Maven

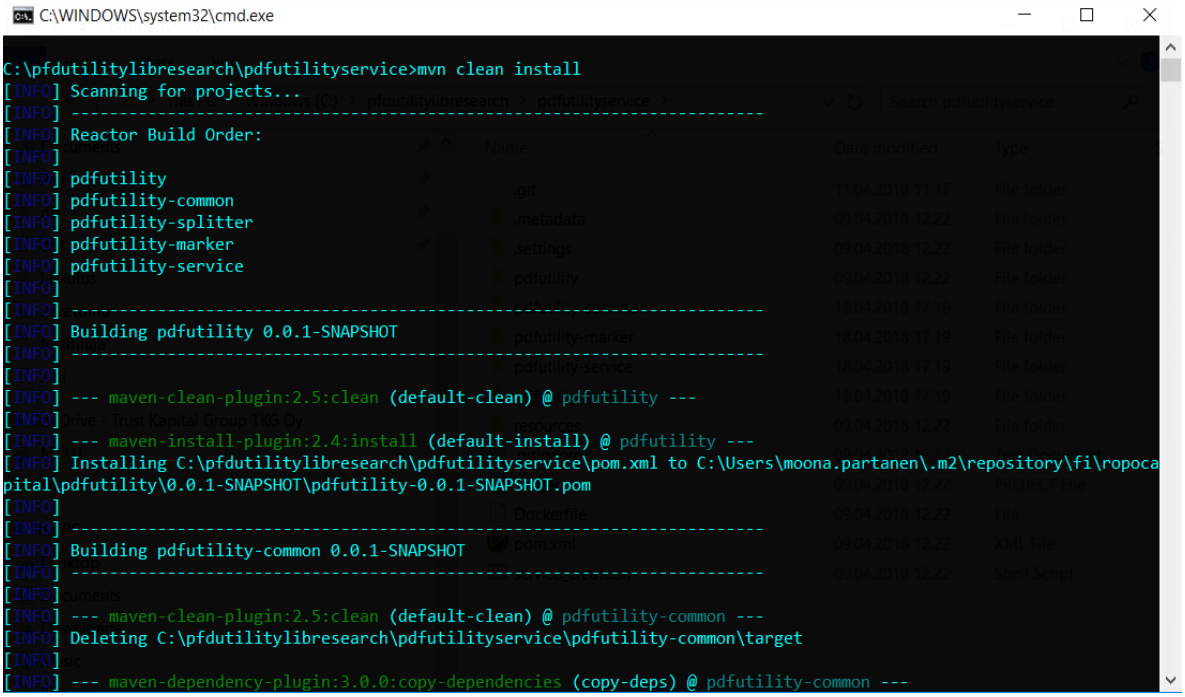
Maven on konsolipohjainen työkalu, jonka tarkoituksena on helpottaa Java-ohjelmien kääntämistä. Mavenin ominaisuuksia ovat muun muassa riippuvuuksien kuten ulkoisten kirjastojen hallinta sekä projektin rakenteen selkeyttäminen. Mavenia käytetään komentokehotteeseen annettavilla käskyillä. (What is Maven? 3.3.2018).

Sovelluksen kehitykseen valittiin aputyökaluksi Maven monesta syystä. Syitä olivat muun muassa Mavenin seuraavat ominaisuudet:

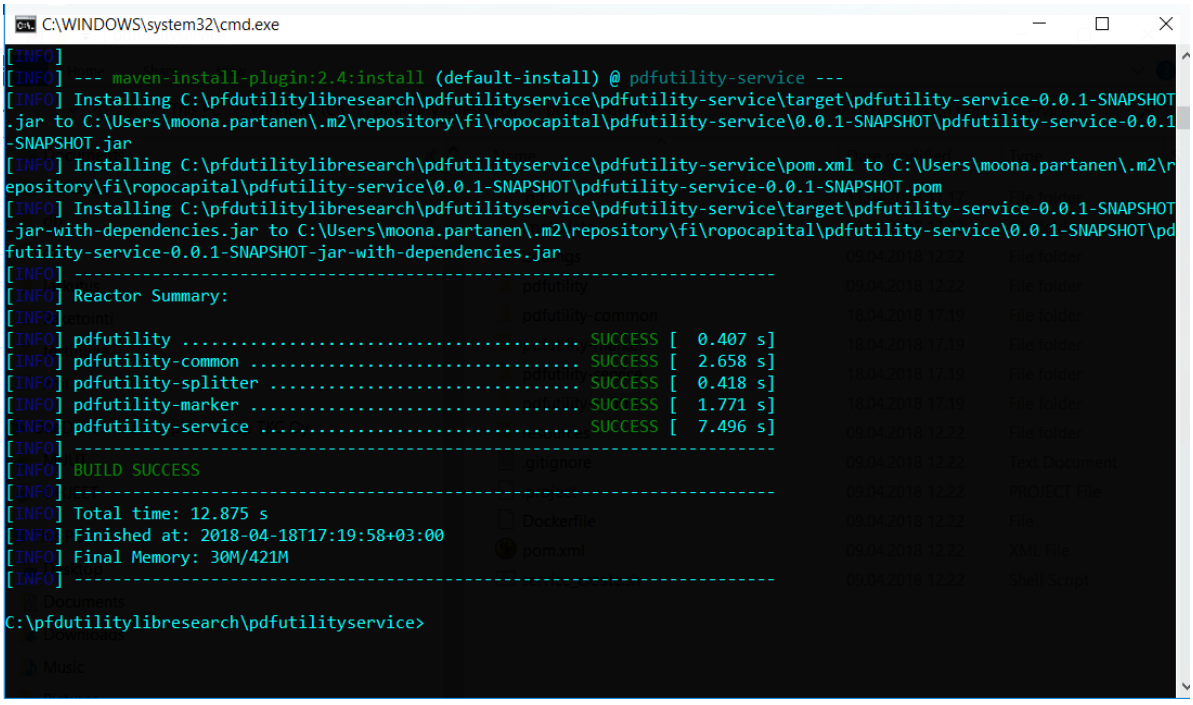
1. Kaikki projektin moduulit voidaan suorittaa vain yhdellä Maven-komennolla
2. Maven huolehtii moduuleiden suoritusjärjestyksestä
3. Maven tekee moduuleista valmiin JAR-paketin, joka sisältää myös kaikki riippuvuudet kuten ulkoiset kirjastot

Maven-projektit sisältävät POM.xml-tiedoston, jossa konfiguroidaan projekti. POM tulee sanoista Project Object Model. POM.xml-tiedostoihin on mahdollista määritellä esimerkiksi ulkoisia kirjastoja, riippuvuuksia sekä liitännäisiä.

Kuvissa 1 ja 2 näkyy, kuinka Maven huolehtii suoritusjärjestyksestä ja rakentaa projektin.



KUVA 1. Maven huolehtii moduuleiden suoritusjärjestyksestä



KUVA 2. Maven rakentaa projektin valmiiksi

3.1.4 Jetty-palvelin

Jetty on Javaa tukeva HTTP-palvelin, jota tavallisesti käytetään koneiden välisiin yhteyksiin. Sovelluksessa päädyttiin käyttämään Jettyä, koska sitä ollaan käytetty Ropo Capitalilla muissakin projekteissa ja se on osoittautunut toimivaksi. (https://www.eclipse.org/jetty/, 3.3.2018).

Sovelluksen kehitysvaiheessa Jetty-palvelimelle konfiguroitiin handlerit eli käsittelijät, jotka käsittelevät Jettylle tulevia pyyntöjä. Kullekkin sovelluksen toiminnolle luotiin oma handler. Jettyä konfiguroitiin myös niin, että palvelinta käynnistettäessä annetaan parametrina portti, jota Jetty sitten kuuntelee. Kuvassa 3 on esitelty startServer-funktio, jossa näkyy hieman Jettyn handlereiden konfigurointia pilkonta- ja merkkaustoiminnoille.

```
public static void startServer(String host, int port)
{
    Server restServer = new Server();
    ServerConnector connector = new ServerConnector(restServer);

    if(host.isEmpty()){
        logger.error("server isn't set your call. Do that");
    }

    connector.setHost(host);
    connector.setPort(port);

    ContextHandler context = new ContextHandler();
    context.setHandler(new DefaultHandler());

    ContextHandler splittercontext = new ContextHandler("/split");
    splittercontext.setHandler(new SplitterHandler());

    ContextHandler markercontext = new ContextHandler("/mark");
    markercontext.setHandler(new MarkerHandler());

    ContextHandlerCollection contexts = new ContextHandlerCollection();
    contexts.setHandlers(new Handler[] { context, splittercontext, markercontext });

    restServer.setHandler(contexts);
    restServer.addConnector(connector);

    try
    {
        restServer.start();
        logger.info("Server listen port " + connector.getPort() + " " + restServer.getURI());
        restServer.join();
    }
    catch (Exception e) {
        logger.error("Problem while starting server, " + e.toString());
    }
}
```

KUVA 3. StartServer-funktio

3.1.5 PDFBox- ja JAXB-kirjastot

Sovellusta kehittäessä hyödynnettiin Apache PDFBox- ja JAXB-kirjastoja. Apache PDFBox on työkalu pdf-tiedostojen manipulointiin. PDFBoxin ominaisuuksia ovat muun muassa tiedostojen luonti, pilkontatyökalu sekä kuvien tai tekstin lisääminen pdf-tiedostoon.

JAXB (Java Architecture for XML Binding) on kirjasto, jolla voidaan käsitellä XML-tiedostoja helposti. JAXB muodostaa XML-tiedostoista generoituja Java-objekteja ja -luokkia tai toisin päin objekteista ja luokista XML-tiedostoja. Generoituihin Java-luokkiin ja -objekteihin tarvitaan skeema XML-tiedostosta.

3.2 Kehitysmenetelmät

3.2.1 Ketterä kehitys ja Scrum

Scrum on projektinhallinnan viitekehys, joka mahdollistaa tehokkaan työskentelyn ketterässä ohjelmistokehityksessä. Scrumissa projektin työstämistä vaiheistetaan sprinteillä eli kehitysjaksoilla, joiden pituus on 1-4 viikkoa. Jokaisen sprintin sisältö sovitaan sprintin suunnittelupalaverissa ennen sprintin aloitusta. (SUTHERLAND, Jeff ja SCHWABER, Ken. Suomenkielinen Scrum Guide. 2012. 13.3.2018)

Ropo Capitalilla sprintit ovat viikon mittaisia jaksoja. Sprinttien edistymistä seurataan viikottaisilla palavereilla sekä päivittäin pidettävillä lyhyillä päiväpalavereilla. Kunkin sprintin lopussa pidetään sprinttikatselmus, jossa käydään yhdessä läpi mitä kukin kehitystiimi on saanut aikaan sprintin aikana.

3.2.2 Yksikkötestit ja sovelluksen testaus

Testauksen tarkoitus on yrittää löytää sovelluksesta mahdollisimman paljon virheitä jo kehitysvaiheessa, jotta tuotantoversioon ei jäisi enää virheitä. Yksikkötestauksella tarkoitetaan sovelluksen sisäisten rakenteiden testausta ja sillä voidaan testata esimerkiksi yksittäisiä luokkia tai niiden metodeja. Sovelluksen yksikkötestit kirjoitettiin niin, että jokaisella moduulilla on omat testiluokkansa. (VESTERHOLM, Mika ja KYPPÖ, Jorma. 2015. Java-ohjelmointi. 9. painos, 3.3.2018).

```
@Test
public void testGetTotalPageCount() {

    assertEquals(fUtility.GetTotalPageCount(testpdf1), 4);
    assertEquals(fUtility.GetTotalPageCount(testpdf2), 252);
}
```

KUVA 4. Esimerkki yksikkötestistä

Kuvassa 4 on esimerkki GetTotalPageCount-metodin yksikkötestistä. Todelliselle GetTotalPageCount-metodille annetaan parametrina pdf-tiedosto ja metodi sitten palauttaa sivumäärän. Testiluokka testGetTotalPageCount vertailee todellisia arvoja oletettuihin arvoihin.

Sovellusta testattiin myös testiympäristössä niin, että se asennettiin tuotannon testipalvelimelle pyörimään Windowsin palveluna (katso kohta 4.4 JAR-paketti Windowsin palveluksi). Näin pystyttiin testaamaan tuotannon ohjelmista tulevien pyyntöjen toimivuus sekä kuormittavuus. Kuormitusta testattiin myös laittamalla sovellus pyörimään AWS:ssa (katso kohta 4.3. AWS), joten pystyttiin testaamaan verkon yli lähetettävien pyyntöjen latenssia.

3.2.3 GIT ja versionhallinta

Sovellusta työstäessä käytettiin apuna GIT-ohjelmistoa. GIT on versionhallintaan käytettävä ohjelmisto, jonka toimintaperiaate on seuraavanlainen:

1. Käyttäjä asentaa omalle tietokoneelleen GITin käyttöliittymän
2. Käyttöliittymässä määritetään palvelimella olevan projektikansion eli repository osoite
3. Käyttöliittymässä haetaan repositoryn työkopio omalle tietokoneelle
4. Työkopioon tehdään muutokset sovellusta kehitettäessä
5. Muutokset lähetetään käyttöliittymän kautta palvelimella sijaitsevaan repositoryyn, jolloin muutokset ovat myös muiden käyttäjien käytettävissä

GITin etuna on helppous työskentelyssä silloin kun sovellusta kehitetään useamman henkilön toimesta. Muita etuja ovat muun muassa se, että kaikki muutokset versioidaan ja voidaan palata takaisin versioissa, mikäli uusin versio on käyttökelvoton.

4 SOVELLUKSEN TOTEUTUS

4.1 Sovelluksen toteutuksesta

Sovelluksen toteusvaiheen sain tehdä melko itsenäisesti, mutta tarvittaessa sain neuvoa muilta kehitystiimin jäseniltä. Toteutin kaikki sovelluksen toiminnot ja ohjelmoinnin, mutta asiakasohjelman ohjelmoinnin hoiti eräs kollegani. Sovellusta kehitettiin pala palalta ja pienissä osissa.

4.2 XML-ohjaustiedostot

Sovellukselle annetaan parametrina xml-muotoinen ohjaustiedosto riippuen toiminnosta, mikä halutaan suorittaa. Ohjaustiedostot kertovat sovellukselle ohjeet siitä, miten toiminto toteutetaan. Esimerkiksi pilkontatoimintoon (pdfutility-splitter-moduuli) annettavassa ohjaustiedostossa on ohjeet siitä, miltä sivulta pilkonta aloitetaan, kuinka monta sivua pilkottuun tiedostoon tulee sekä minkä nimiseksi pilkottu tiedosto nimetään. Kuvassa 5 on esimerkki pilkontamoduuliin annettavasta ohjaustiedostosta.

```
<?xml version="1.0" encoding="utf-8"?>
<Documents>
  <Part outputname="splittedDocument1.pdf">
    <Startpage>3</Startpage>
    <Pagecount>57</Pagecount>
  </Part>
  <Part outputname="splittedDocument2.pdf">
    <Startpage>7</Startpage>
    <Pagecount>12</Pagecount>
  </Part>
</Documents>
```

KUVA 5. Esimerkki ohjaustiedostosta

Ohjaustiedostoille luotiin XSD-rakennekuvaukset eli skeemat. Syitä, miksi skeemat luotiin olivat:

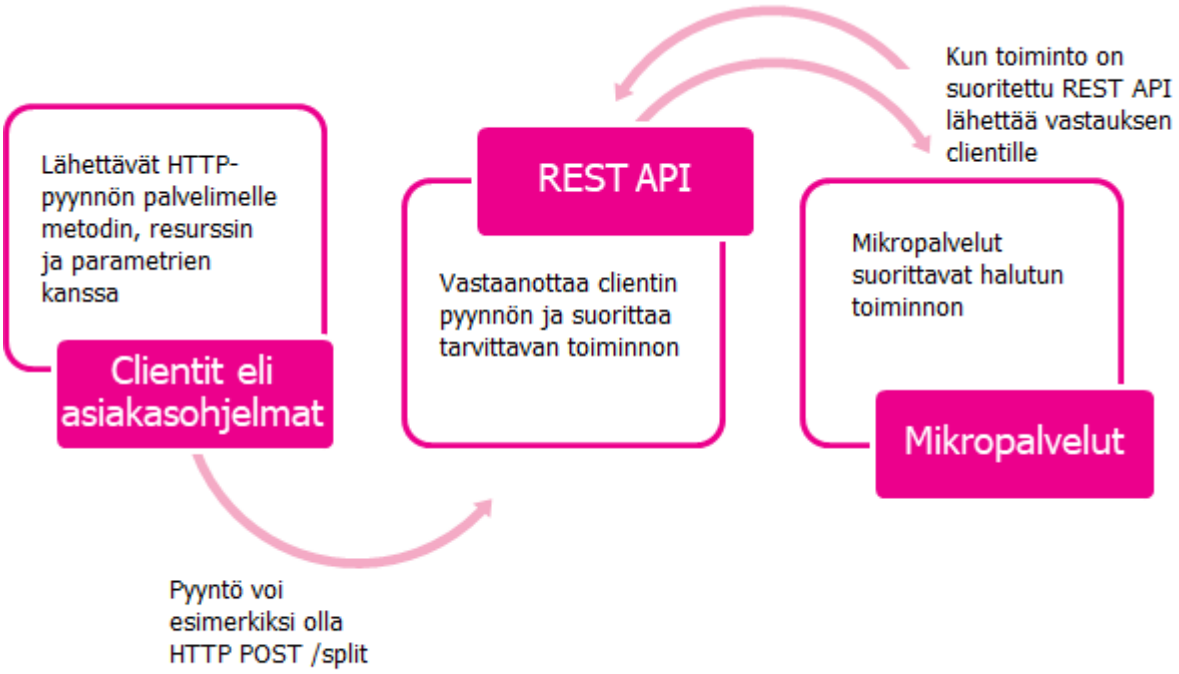
1. Yhteisen sanaston rakentaminen
2. Skeema toimii sääntönä, joita ohjaustiedoston tulee noudattaa
3. Skeemojen avulla voidaan helposti tehdä generoituja luokkia Ecplipsessä JAXB: n avulla
4. Sovelluksen dokumentointia varten (katso kohta 4.7 Dokumentointi)

4.3 REST API

REST tulee sanoista Representational State Transfer ja se tarkoittaa HTTP-protokollaan perustuvaa arkkitehtuurimallia ohjelmointirajapintojen toteuttamiseen. REST-arkkitehtuurin on tarkoitus parantaa rajapintojen suorituskykyä ja yksinkertaisuutta. REST-rajapinnan ideana on, että asiakasohjelma (client) lähettää pyynnön palvelimelle, joka sitten käsittelee pyynnön ja palauttaa vastauksen asiakasohjelmalle. Pyynnöt ovat tarkasti määriteltyjä ja ne sisältävät metodin, jota halutaan käyttää, resurssin, josta tietoa haetaan sekä mahdolliset parametrit.

HTTP-protokollan tarjoamia yleisimpiä metodeja ovat GET, POST, PUT ja DELETE. Tässä sovelluksessa kuitenkin käytetään lähinnä vain POST-metodia. Pyynnön mukana lähetetään myös resurssi, joka määrää mikä toiminto palvelimella suoritetaan. Tämän sovelluksen useimmissa pyynnöissä asiakasohjelman tulee lähettää myös parametrit.

Kaaviossa 2 on kuvattu REST APIn toimintaa sovelluksessa.



KAAVIO 2. REST APIn toiminta sovelluksessa

Koska sovelluksella on useita toimintoja, clientin pyynnöt ovat erilaisia halutusta toiminnosta riippuen. Taulukossa 2 on kuvattu erilaisia tarkoin määritettyjä pyyntöjä, joita client voi lähettää palvelimelle.

Haluttu toiminto	Resurssi	Parametrit	Vastaus
Ison pdf-tiedoston pilkonta pienempiin osiin	/split	Pdf-tiedosto sekä xml-muotoinen ohjaustiedosto	Zip-paketti, joka sisältää pilkotut tiedostot
Postin palautuskoodin merkkaus pdf-tiedoston tietyille sivuille ja tiettyyn positioon	/mark	Pdf-tiedosto sekä xml-muotoinen ohjaustiedosto	Pdf-tiedosto, jossa halutuille sivuille on merkattu palautuskoodi
Tiedoston konvertointi	/convert	Tiedosto, joka halutaan konvertoida sekä tiedostomuoto, johon tiedosto halutaan muuttaa. Esimerkiksi Excel-tiedosto, joka halutaan konvertoida pdf-tiedostoksi.	Konvertoitu tiedosto
Pdf-tiedostojen yhdistely yhdeksi pdf-tiedostoksi	/concatenate	Zip-tiedosto, joka sisältää yhdisteltävät pdf-dokumentit sekä xml-muotoinen ohjaustiedosto	Pdf-tiedosto, johon on yhdistelty dokumentit halutussa järjestyksessä

TAULUKKO 2. Esimerkkipyyntöjä

4.4 AWS

AWS eli Amazon Web Services on laajasti käytetty pilvipalvelualusta erilaisille digitaalisille palveluille. Sitä hyödyntää esimerkiksi Netflix. AWS tarjoaa runsaasti ominaisuuksia skaalautuvien, tietoturvallisten ja liiketoimintakriittisten digitaalisten palveluiden rakentamiseen. (https://www.nebula.fi/ict-ratkaisut/digitaalinen-liiketoiminta/pilvialustat/amazon-web-services/, 11.3.2018)

Testausvaiheessa sovellus laitettiin pyörimään AWS: lle testaaksemme verkon yli tulevien pyyntöjen latenssia. Sovellus pyörii AWS: ssa t2. micro-instanssilla, joka käytännössä on virtuaalinen Linux-palvelin. Pyyntöjä voidaan lähettää sovellukselle samalla tavalla kuin yllämainitussa (Katso kohta 4.3 REST API), sillä Jetty-palvelimeen on konfiguroitu erikseen handler AWS: n kautta tuleville pyynnöille.

Testausvaiheessa AWS: lla havaittiin pieniä muisti- ja latenssiongelmia. Isojen aineistojen testauksessa huomattiin, että verkon yli tulevissa pyynnöissä asiakasohjelma joutuu odottelemaan vastausta melko kauan. Muistiongelmat johtuivat sovelluksessa käytetystä PDFBox-kirjastosta (katso kohta 3.1.5 PDFBox- ja JAXB-kirjastot), joka jätti paljon turhia resursseja auki sovellukseen.

4.5 JAR-paketti Windowsin palveluksi

Testausvaiheessa sovellus asennettiin pyörimään tuotannon testipalvelimelle Windows-palveluna. Asentamiseen Windows-palveluksi päädyttiin siksi, että sovellus pyörisi palveluna jatkuvasti, eikä sitä tarvitsisi erikseen käynnistää. JAR-paketin asentamisessa Windows-palveluksi käytettiin apuna Mavenia sekä Apache Common Daemon-kirjastoa.

Asentaminen aloitettiin konfiguroimalla pdfutility-service-moduulin POM.xml-tiedostoa niin, että kaikki tarvittavat komponentit kopioidaan yhteen kohdehakemistoon, joka nimetään nimellä Installer. Tarvittavia komponentteja ovat JAR-tiedosto riippuvuuksineen, JRE (Java Runtime Environment), commons-daemonin binääritiedostot sekä install.bat-tiedosto asennusta varten. Kuvassa 6 on kuvattu Maven pluginin konfigurointia pdfutility-service-moduulin POM.xml-tiedostossa.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-antrun-plugin</artifactId>
    <version>1.7</version>
    <executions>
        <execution>
            <id>default-cli</id>
            <phase>package</phase>
            <goals>
                <goal>run</goal>
            </goals>
            <configuration>
                <target>
                    <copy todir="${installer.dir}/jre1.8.0_91">
                        <fileset dir="${project.basedir}/jre1.8.0_91" />
                    </copy>
                    <copy todir="${installer.dir}/commons-daemon">
                        <fileset dir="${project.basedir}/commons-daemon" />
                    </copy>
                    <copy file="${project.build.directory}/pdfutility-service-0.0.1-SNAPSHOT-jar-with-dependencies.jar"
                          todir="${installer.dir}" />
                    <copy file="${project.basedir}/install.bat"
                          todir="${installer.dir}" />
                    <copy file="${project.basedir}/uninstall.bat"
                          todir="${installer.dir}" />
                    <copy file="${project.basedir}/LICENSE"
                          todir="${installer.dir}" />
                </target>
            </configuration>
        </execution>
    </executions>
</plugin>
```

KUVA 6. Maven pluginin konfigurointia

Install.bat-tiedosto käyttää commons-daemon-kirjastoa asentaakseen sovelluksen Windows-palveluksi. Install.bat-tiedostoon määriteltiin muun muassa metodit, joilla hallitaan käynnissä olevaa sovellusta. Kuvassa 7 on kuvattu install.bat-tiedoston sisältö.

```
commons-daemon\prunsrv //IS//Pdfutility
--DisplayName="Pdfutility"
--Description="Pdfutility"
--Install="%cd%\commons-daemon\prunsrv.exe"
--Jvm="%cd%\jre1.8.0_144\bin\client\jvm.dll"
--StartMode=jvm
--StopMode=jvm
--Startup=auto
--StartClass=fi.ropocapital.pdfutility.service.RestServer
--StopClass=fi.ropocapital.pdfutility.service.RestServer
--StartParams=start
--StopParams=stop
--StartMethod=windowsService
--StopMethod=windowsService
--Classpath="%cd%\pdfutility-service-0.0.1-SNAPSHOT-jar-with-dependencies.jar"
commons-daemon\prunsrv //ES//Pdfutility
```

KUVA 7. Install.bat-tiedoston sisältö

Kun sovellus ajetaan Mavenilla, syntyy Installer-hakemisto, joka sisältää JAR-tiedoston, JRE: n, commons-daemonin sekä install.bat-tiedoston. Installer-hakemisto voidaan sitten pakata SFX-tiedostoksi (self-extractable archive).

4.6 Esimerkki clientista

Sovellusta voidaan kutsua eri asiakasohjelmista. Sovellusta kutsutaan Ropo Capitalilla kuitenkin suurimmaksi osaksi Perl-ohjelmointikielellä ohjelmoidusta clientista.

Client tekee ensin halutun xml-muotoisen ohjaustiedoston ja sen jälkeen lähettää pyynnön sovellukselle parametreineen (katso kohta 4.3 REST API). Kun client saa vastauksena tiedoston riippuen pyynnöstä, vastaus parsitaan ja käsittelyä jatketaan. Jos pyyntö jostain syystä ei onnistu, client kirjoittaa virheviestin tuotannon lokille, jotta mahdollinen ongelma on helpompi selvittää.

Kuvassa 8 on esimerkki Perl-ohjelmointikielellä toteutetusta clientista, jossa näkyy kuinka sovellukselle lähetetään pyyntö.

```
kirjoita_loki("*** Split large $PDFFilename to individual files ***",$0);
$userAgentObject = LWP::UserAgent->new();

##
## Request - via hash-table
##
print $SERVER_URI;

$response = $userAgentObject->post($SERVER_URI,
                                   'Content_type' => "form-data",
                                   'Content' => [
                                                 'pdfFileToUpload'   => [$directory.$PDFFilename,$PDFFilename, 'Content_Type=>application/pdf'],
                                                 'xmlFileToUpload'   => [$directory.$xmlFilename,$xmlFilename, 'Content_Type=>text/xml']
                                                ],
                                   ':content_cb'        => \&pdfUtility::byteSave,
                                   ':read_size_hint'    => 150000,
                                  );
##
## Here is some checkings about response. We want to check, that we get right information and can continue
##

$responseMessage    = $response->message();
$responseSize       = $response->headers->content_length();
$responseContent    = $response->content();

kirjoita_loki("*** Response message: ".$responseMessage." ***",$0);

virhe("*** Response error: ".$response->status_line." ***",$0) unless($response->is_success);
virhe("*** Response file size is too small: ".$responseSize." ***",$0) if($responseSize < 10);
```

KUVA 8. Esimerkki Perl-clientista

4.7 Dokumentointi

Sovelluksen palvelujen toiminta, ylläpito-ohjeet sekä ohjeistus häiriötilanteisiin tullaan dokumentoimaan Ropo Capitalin sisäiseen intranettiin. Dokumentointi on tärkeää, sillä mahdollisten häiriötilanteiden selvitys on helpompaa dokumentaation avulla. Sovellukseen liittyvissä asioissa kuten esimerkiksi päivitys- tai huoltokatkotilanteissa myös tiedotetaan muita tiimejä ja työntekijöitä sähköpostilla.

Dokumentaatiossa kerrotaan myös sovellukselle mahdollisten pyyntöjen parametreista ja resursseista. Myös parametrina annettavien ohjaustiedostojen skeemat dokumentoidaan, sillä kun sovellukselle halutaan lähettää pyyntöjä uudesta asiakasohjelmasta, ohjelman kehittäjä voi helposti käydä katsomassa intrasta, minkälainen pyynnön tulee olla ja mitä parametreja sen mukana lähetetään. Tämä helpottaa huomattavasti niiden uusien ohjelmien kehitystä, joissa tarvittaan pdf-tiedostojen käsittelyyn liittyviä toimintoja.

5 YHTEENVETO JA POHDINTA

Opinnäytetyöni tarkoituksena oli suunnitella ja kehittää sovellus, joka suorittaa aiemmin Inspiren toteuttamia toimintoja pdf-tiedostojen käsittelyssä ja muokkauksessa. Sovelluksen toteutuksessa oli myös reunaehtoja, kuten se, että sitä voidaan kutsua monesta eri asiakasohjelmasta sekä sovelluksen toimivuus isojen aineistojen kanssa. Lopputuotteena syntyi http-palvelu, joka otettiin heti käyttöön.

Opinnäytetyötäni aloittaessa sovimme tavoitteeksi saada valmiiksi ainakin pilkontatoiminto, jolla oli todellinen tarve Ropo Capitalin aineistokäsittelyssä. Pilkontatoiminto oli siis opinnäytetyöni tärkein ominaisuus. Tavoitteisiin päästiin ja sain valmiiksi myös muita ominaisuuksia. Sovellusta tullaan kuitenkin vielä jatkokehittämään, mikäli Inspirestä luopumisessa tulee ilmi vielä toimintoja, jotka liittyvät pdf-tiedostojen käsittelyyn.

Opinnäytetyötäni työstäessä ongelmia aiheuttivat osittain ajan puute. Koska olen vielä tällä hetkellä opiskelija ja minulla on luentoja sekä tunteja koululla, olin fyysisesti paikalla Ropo Capitalilla vain kolmena päivänä viikossa. Lisäksi Inspiren alasajoprojektin takia muut kehitystiimiläiset olivat hyvin kiireisiä, joten yhteistä aikaa ongelmien ilmetessä ja perehdytykseen oli hieman hankala löytää. Työstinkin opinnäytetyöni ohjelmointiosuutta paljon vapaa-ajallani ja välillä tuntuikin, että vapaa-aikaa ei jää ollenkaan.

Toinen haasteita aiheuttanut ongelma opinnäytetyöni toteutusvaiheessa oli PDFBox-kirjaston muistiongelmat. PDFBox-kirjasto on muuten erittäin kätevä, mutta luulen, ettei sitä ole tarkoitettu kovin isojen aineistojen käsittelyyn. Jatkokehityssuunnitelmana sovellukselle onkin muuttaa PDFBox-kirjasto johonkin muuhun pdf-tiedostoja käsittelevään ja muokkaavaan kirjastoon mikäli muistiongelmia tulee vielä ilmi. Jos olisin tiennyt PFDBoxin muistiongelmista etukäteen, olisin valinnut alunperinkin jonkun muun kirjaston.

Pieniä haasteita aiheutti myös opinnäytetyöni alkuvaiheessa Maven, koska en ollut aiemmin käyttänyt sitä. Mavenin käytössä ongelmat johtuivat POM.xml-tiedostoista, koska en ollut tottunut käyttämään niitä ja välillä konfiguroinnit olivat väärin. Väärät konfiguroinnit aiheuttivat virheitä ja minulla meni aikaa virheiden selvittämiseen. Kuitenkin nyt kun olen oppinut käyttämään Mavenia, pidän sitä erittäin hyvänä työkaluna ja aion käyttää sitä jatkossa muissakin Java-projekteissa.

Opinnäytetyötäni tehdessä sain itsenäisesti kehittää koko sovelluksen ja opin todella paljon uutta ohjelmoinnista, erilaisista tekniikoista ja ennen kaikkea sovelluksen arkkitehtuurista. Sain suunnitteluvaiheessa vinkkejä siihen, minkälainen rakenne olisi järkevä toteuttamaani sovellukseen ja pidinkin vinkkejä todella arvokkaina.

Kokonaisuudessaan opinnäytetyöni suunnittelu- ja toteutusvaihe sujui mielestäni oikein hyvin, sillä asetetut tavoitteet täyttyivät ja sovellukseen tuli myös ominaisuuksia, jotka oltiin ajateltu toteuttaa vasta myöhemmässä vaiheessa. Pääsin myös työskentelemään osana kehitystiimiä ja sain kehittää taitojani ohjelmoijana. Koska sovellus otettiin käyttöön Ropo Capitalilla, olen myös erittäin tyytyväinen siihen, että sain tehdä opinnäytetyönäni niin sanotusti oikean ohjelman oikeisiin tarpeisiin.

LÄHTEET

VESTERHOLM, Mika ja KYPPÖ, Jorma. 2015. Java-ohjelmointi. 9. painos. Helsinki: Talentum. (luettu 3.3.2018). (Viitattu 3.3.2018).

ECPLISE WIKI. Simultaneous Release. (luettu 3.3.2018). Saatavissa:
http://wiki.eclipse.org/Simultaneous_Release

MAVEN. What is Maven? (luettu 3.3.2018). Saatavissa:
https://maven.apache.org/what-is-maven.html

Comparing XML Schema Languages, XML.com (luettu 3.3.2018). Saatavissa:
http://www.xml.com/pub/a/2001/12/12/schemacompare.html

PDFBox, Apache (luettu 11.3.2018). Saatavissa:
https://pdfbox.apache.org/

SUTHERLAND, Jeff ja SCHWABER, Ken. Suomenkielinen Scrum Guide. 2012. Verkkojulkaisu. (luettu 13.3.2018). (viitattu 13.3.2018). Saatavissa:
https://scrumwell.files.wordpress.com/2012/01/2016-scrum-guide-fi.pdf

PELTONEN, Janne, Inspiren alasajo ehdotus. Ropo Capital. Verkkojulkaisu. (luettu 7.3.2018). Saatavissa: Ropo Capital Intranet

KANKAANPÄÄ, Sami. REST-arkkitehtuurityylin käyttö web-rajapinnoissa. Opinnäytetyö. (luettu 13.3.2018). Saatavissa:
https://www.theseus.fi/bitstream/handle/10024/113311/Kankaanpaa_Sami.pdf?sequence=1&isAllowed=y

ORT, Ed ja MEHTA Bhakti. 2003. Java Architecture for XML Binding (JAXB). Verkkojulkaisu. (luettu 12.3.2018). Saatavissa:
http://www.oracle.com/technetwork/articles/javase/index-140168.html#

What is Git? Video. (katsottu 7.3.2018) Saatavissa:
https://git-scm.com/video/what-is-git

Amazon Web Service. (luettu 11.3.2018). (viitattu 11.3.2018). Saatavissa:
https://www.nebula.fi/ict-ratkaisut/digitaalinen-liiketoiminta/pilvialustat/amazon-web-services

What is AWS? Video. (katsottu 11.3.2018). Saatavissa:
https://aws.amazon.com/what-is-aws/

MAVEN. POM Reference. (luettu 13.3.2018). Saatavissa:
https://maven.apache.org/pom.html#What_is_the_POM

Design Patterns. (luettu 29.3.2018). Saatavissa:
https://sourcemaking.com/design_patterns

Sähköpostikeskustelu ohjelmistokehittäjä Simo Hiltusen kanssa koskien sovelluksen arkkitehtuuria. (sähköposti vastaanotettu 22.2.2018).

Extensible Markup Language (XML). Verkkojulkaisu. (luettu 8.4.2018). Saatavissa:
https://www.w3.org/XML/

Mikropalvelut nousivat hypen huipulle – mitä hyötyä niistä on? Verkkojulkaisu. (luettu 3.3.2018). Saatavissa:
https://www.tivi.fi/Kaikki_uutiset/mikropalvelut-nousivat-hypen-huipulle-mita-hyotya-niista-on-6534379

Tarvitaanko mikropalveluja oikeasti? Verkkojulkaisu. (luettu 3.3.2018). Saatavissa: https://www.mystes.fi/tarvitaanko-mikropalveluja-oikeasti/

API:en hallinta. Verkkojulkaisu. (luettu 3.3.2018). Saatavissa: https://www.solita.fi/apien-hallinta/

Parent POMs. Verkkojulkaisu. (luettu 3.3.2018) Saatavissa: https://maven.apache.org/pom/index.html

Jetty. (luettu 3.3.2018). Saatavissa: https://www.eclipse.org/jetty/